Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 **Jiří Krasula**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Krasula**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: E LINKX a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Dr. Ing. Eduard Sojka**

Konzultant bakalářské práce: Ing. Roman Hrdý

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *30. dubna 2018*

.....

Poděkování

Rád bych poděkoval Ing. Romanovi Hrdému za možnost vykonat ve firmě bakalářskou praxi, Ing. Soni Činčalové za vedení a dohled nad projekty, Petrovi Kubíkovi za odbornou pomoc a cenné rady. Děkuji také ERP a webovému oddělení za konzultace a doc. Dr. Ing. Eduardu Sojkovi za vedení a pomoc se zpracováním bakalářské práce.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava."

V Ostravě dne: 27. dubna 2018

ELINKX a.s. elinkx
Novoveská 1262/95
709 00 Ostrava-Mariánské Hory
IČ: 25847180 DIČ: CZ25847180 (03)

Abstrakt

Cílem této bakalářské práce je popsat mé působení ve společnosti E LINKX a. s., ve které jsem byl přidělen na Enterprise Resource Planning (ERP) & web oddělení. Stěžejním bodem práce je projekt napojení ERP systému Esyco .NET na službu MailChimp skrze rozhraní API. Jelikož systémů na e-mail marketing je opravdu mnoho, jako další projekt je i napojení na službu SmartEmailing. V neposlední řadě také popíšu testování API a prvotní vývoj automatizovaných testů, se kterými jsem neměl do této doby zkušenost. V této práci popíšu tyto projekty, jejich problematiku a samotný postup řešení.

Klíčová slova: Bakalářská práce; odborná praxe; E LINKX a. s.; API; automatizované testování API; SoapUI; JSON; ERP systém; Esyco .NET; C#; .NET Framework; MailChimp; SmartEmailing

Abstract

The aim of this bachelor thesis is to describe my work at the E LINKX a. s., in which I was assigned to the Enterprise Resource Planning (ERP) & web department. The primary point of work is the project about connecting the ERP system Esyco .NET to MailChimp service through the API interface. Since there are really many e-mail marketing systems, the next project is a connection to SmartEmailing service. Finally, I will also describe the API testing and the initial development of automated tests with which I have not had experience so far. In this thesis I will describe these projects, their problems and the process of solving itself.

Key words: Bachelor thesis; professional experience; E LINKX a. s.; API; automated API testing; SoapUI; JSON; ERP system; Esyco .NET; C#; .NET Framework; MailChimp; SmartEmailing

Obsah

Seznam použitých symbolů a zkratek

API	– Application Programming Interface
B2B	– Business-to-business
B2C	– Business-to-consumer
CRM	– Customer relationship management
CRUD	– Create, Read, Update, Delete
ERP	– Enterprise Resource Planning
GUI	– Graphical User Interface
HTTP	– Hypertext Transfer Protocol
JSON	– JavaScript Object Notation
MVC	– Model-view-controller
REST	– Representational State Transfer
RPC	– Remote procedure call
SOAP	– Simple Object Access Protocol
SQL	– Structured Query Language
SSMS	– SQL Server Management Studio
TFS	– Team Foundation Server
T-SQL	– Transact-SQL
XML	– eXtensible Markup Language

Seznam ilustrací a seznam tabulek

Úvod

Tématem bakalářské práce je popsat mé působení ve společnosti E LINKX a. s., ve které jsem byl přidělen do ERP & web oddělení. Hlavním důvodem absolvování bakalářské práce formou odborné praxe ve firmě byla vize pracovat v týmu na reálných projektech, které mi poskytnou cenné zkušenosti, které budu moct v budoucnu dále uplatnit. Dále také získat přehled o pracovních pozicích, abych zjistil, co přesně bych chtěl v budoucnu dělat.

Důvodem, proč jsem si vybral tuto společnost bylo, že již v předchozím ročníku jsem zde vykonával předmět Praxe, díky kterému jsem získal nejen zkušenosti, ale i pracovní místo na pozici testera. O společnosti jsem se poprvé doslechl na pracovním veletrhu Kariéra PLUS a v první kapitolu věnuji právě jejímu představení a zaměření. V dalších sekcích popisuji své pracovní zařazení, náplň a prostředí.

V kapitole 3 je orientační časový harmonogram projektů, na kterých jsem pracoval. Testování API rozhraní, které mi bylo přiděleno na začátku praxe, popisuji v kapitole 4. V další kapitole se věnuji prvotnímu vývoji automatizovaných testů pro API rozhraní, se kterými jsem neměl do této doby žádnou zkušenost.

Stěžejnímu bodu práce projektu MailChimp, jehož cílem je napojení ERP systému Esyco .NET na službu MailChimp skrze rozhraní API, se věnuji v kapitole 6. Tam popíšu jak samotnou implementaci, tak i její návrh. Jelikož systémů na e-mail marketing je opravdu mnoho, v další kapitole se věnuji projektu SmartEmailing, kde mým cílem byla analytická činnost. V poslední kapitole shrnuji dosažené výsledky, získané zkušenosti a celkově hodnotím odbornou praxi, kterou jsem měl tu čest absolvovat.

1 Popis odborného zaměření firmy a pracovního zařazení

1.1 O firmě

E LINKX a. s. [1] je ryze česká společnost, která působí na trhu již více než 15 let a je partnerem skupiny eD‘ system Group. Společnost je systémovým integrátorem, což vychází z rozsáhlého portfolia vlastních aplikací, produktů třetích stran a strategických partnerství s významnými subjekty na trhu informačních technologií.

Vyvíjí a realizuje podnikové informační systémy (ERP), prostřednictvím nichž může řešit požadavky zákazníků z různých oblastí výroby, obchodu či služeb. Mezi jejich hlavní produkty patří:

- esyco business – ekonomický informační systém ERP, CRM
- eLogis WMS – systém pro řízení skladů, e-commerce B2B, B2C
- Interlink – internetový obchod na míru propojitelný s esyco business
- eTransys – přepravně logistický informační systém

Společnost je členem Microsoft Partner Network a dlouholetým držitelem kompetence Gold Hosting. Mezi významné společnosti, se kterými firma spolupracuje, patří Hewlett-Packard, Zebra, IBM či Oracle. Ve spolupráci s HP provozuje originální internetový obchod HPmarket.cz.

1.2 Pracovní zařazení a pracovní náplň

Ve společnosti jsem ze začátku pracoval jako tester/web tester. Mou hlavní náplní v té době bylo testování pro API rozhraní pomocí aplikace SoapUI. Se vzrůstajícím počtem nových metod a zákazníků jsem dostal za úkol prvotní vývoj automatizovaných testů.

Později jsem se připojil k projektu MailChimp, kde mou hlavní úlohou byla implementace. Nejdříve jsem však pracoval na části analýzy, poté na návrhu řešení a samotné implementaci. Součástí taktéž bylo vypracování technické dokumentace, uživatelského manuálu a nasazení do testovacího provozu.

Nakonec jsem si vyzkoušel i čistě analytickou činnost u dalšího projektu s názvem SmartEmailing. Na tomto projektu jsem dělal především analytickou činnost, aktuálně testování či uživatelský manuál.

1.3 Pracovní prostředí

E LINKX a. s. sídlí v Ostravě, kde má i jednu ze svých poboček. Druhá pobočka se nachází v Praze. Kanceláře jsou rozděleny na dvě hlavní oddělení a to ERP & web a Logistika. Tyto oddělení se pak dále dělí na divize, kde v každé divizi je průměrně pět lidí. Každý zaměstnanec má k dispozici své místo vybavené stolním počítačem. V případě potřeby je mu ještě přidělen notebook.

Komunikace v rámci projektů, požadavků či úkolů probíhá pomocí aplikace HelpDesk [8], a to jak interně, tak i externě přímo se zákazníky. Pro vnitřní komunikaci je používán emailový klient nebo Skype pro společnosti.

2 Harmonogram bakalářské práce

1. - 2. týden **Testování API rozhraní**

Bylo mi předáno testování API rozhraní pomocí aplikace SoapUI. Mým úkolem bylo nastudovat si dokumentaci ke všem již existujícím metodám, seznámit se s aplikací, vyzkoušet si volání těchto metod a ověřit správnost dat. Následně jsem testoval nově vzniklé metody a dále rozšiřoval dokumentaci.

3. - 6. týden **Vývoj automatizovaných testů API rozhraní**

Vedoucím webového oddělení mi byla předána vize o automatizovaných testech. Automatizované testy jsem tvořil za běhu, jak pro nově vzniklé metody, tak pro starší metody, které bylo třeba testovat pro nové zákazníky.

Také nám byl zadán projekt MailChimp, u kterého začala kolegyně pracovat na analýze. Já jsem se zúčastnil pouze konzultací, abych byl v obraze pro další postup.

7. - 10. týden **Analýza a návrh implementace MailChimp**

Dále jsem se připojil k projektu MailChimp, který už měl téměř hotovou analytickou část. Nejdříve jsem pracoval na části analýzy, poté na návrhu řešení. Implementaci jsme se rozhodli rozdělit na tři hlavní fáze, kdy každá z nich měla být na sobě nezávislá. Projekt se tedy rozdělil na menší bloky, pro které jsem stanovil časovou náročnost na implementaci.

10. - 15. týden **Implementace MailChimp – 1. fáze**

Úkolem 1. fáze bylo naimplementovat základní strukturu aplikace a REST API službu pro komunikaci s API MailChimpu. Dalším krokem byl import a zajištění synchronizace klientů (subjektů) se službou MailChimp. Ten zahrnoval implementaci nových metod REST API služby, přípravu dat i samotný proces synchronizace.

16. - 18. týden **Implementace MailChimp – 2. fáze**

Úkolem 2. fáze byl import a zajištění synchronizace distribučních seznamů a marketingových zařazení se službou MailChimp, včetně segmentování klientů (subjektů) z předchozí fáze. V této fázi se zároveň řešilo i zpracování odpovědí z API MailChimpu.

19. - 23. týden **Implementace MailChimp – 3. fáze**

Úkolem 3. fáze byl import a zajištění synchronizace produktů se službou MailChimp. Tato fáze však zahrnovala nejen produkty, ale i jejich regiony, do kterých jsou produkty rozděleny.

24. týden **Dokončovací práce MailChimp**

Cílem bylo finální dokončení aplikace. Důraz se kladl hlavně na komentáře v kódu. Taktéž bylo třeba vypracování technické dokumentace, uživatelského manuálu a nasazení do testovacího provozu.

25. - 27. týden **Analýza SmartEmailing**

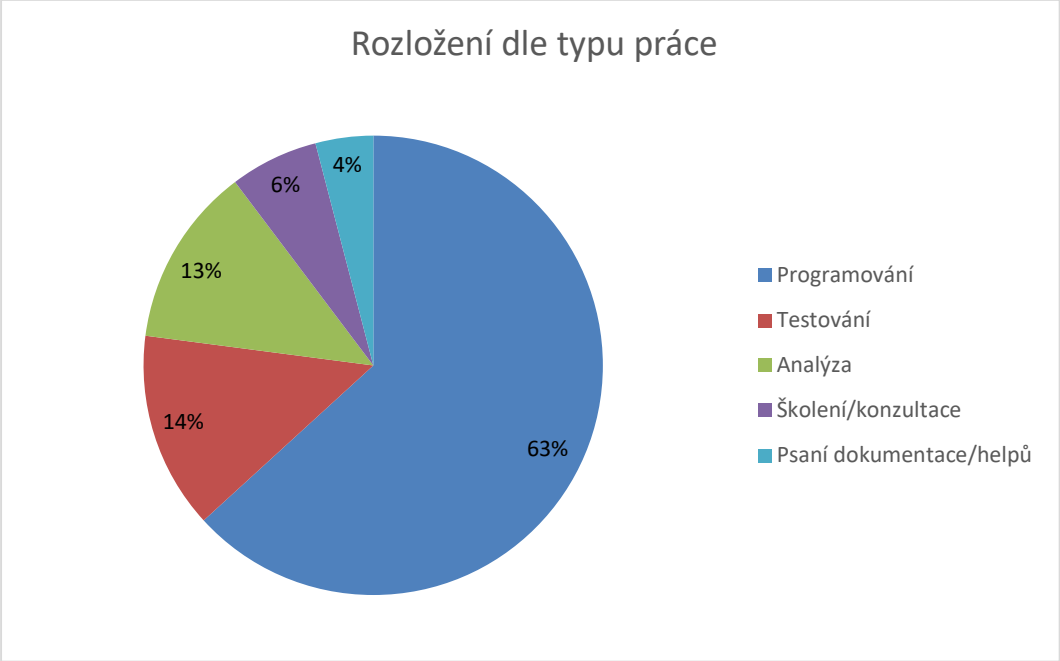
Dalším a zároveň posledním projektem v rámci odborné praxe byl SmartEmailing, tedy další systém pro e-mail marketing. S kolegyní jsme si však měli prohodit role. Začal jsem tedy pracovat na komplexní analýze k tomuto projektu. Aktuálně již řeším testování a uživatelský manuál tak, aby bylo možné aplikaci nasadit do testovacího provozu.

2.1 Přehled odpracovaných hodin

Během vykonávání odborné praxe jsem pracoval na několika projektech, přičemž celkově jsem odpracoval 415,6 hodin.

Na obrázku 1.1 lze vidět jednotlivé rozložení dle typu práce z čehož lze usoudit, že nejvíce času jsem strávil programováním, dále pak testovací či analytickou činností. Přehled odpracovaných hodin v rámci každého projektu uvedu samostatně v každé kapitolách každého z projektů.

Práce	Hodin celkem
Testování API rozhraní	36,1
Automatizace testů API rozhraní	58,9
Projekt MailChimp	274,2
Projekt SmartEmailing	37,7



Obrázek 1.1: *1 Poměr odpracovaných hodin k typu prací*

3 Použité nástroje a technologie

Mou praxi si nedokážu představit bez použití níže uvedených nástrojů, rámců a technologií. Nejdůležitějším a nejpoužívanějším nástrojem během odborné praxe pro mne bylo Visual Studio 2017 s rozšířením ReSharper, ve které probíhal vývoj aplikace v C#, a také jsem si v tomto prostředí osvojil práci s TFS. Pro práci s databází jsem využíval SSMS a pomocí jazyka T-SQL jsem tvořil tabulky, funkce i procedury, ale také mnohdy obtížnější dotazy pro získání přesných výsledků.

3.1 C#

C# [2] je objektově orientovaný programovací jazyk firmy Microsoft stejně jako platforma .NET Framework. V roce 1999 Anders Hejlsberg sestavil tým s účelem vybudovat nový jazyk, který byl o rok později představen pod svým názvem C# společně s projektem .NET. Aktuální verze je 7.2 jež je součástí novější verze Visual Studia 2017, který podporuje i nejnovější .NET Framework 4.7.1.

3.2 .NET a .NET Framework

.NET [3] je multiplatformní softwarová platforma zastřešující portfolio firmy Microsoft, která byla vydaná jako reakce na platformu Java. Základní komponentou je .NET Framework, který nabízí aplikacím potřebné prostředí i knihovny.

Microsoft .NET Framework [4] je vývojová platforma pro vývoj aplikací a služeb, a také je již nedílnou součástí operačního systému Windows. Součástí .NET Frameworku je například i technologie ASP.NET pro vývoj webových aplikací [5] [6].

3.3 Esyco .NET

Esyco .NET [7] je všestranný ekonomický informační systém, jež umožnuje spravovat z jednoho místa e-shop a také řídit skladové zásoby spolu s vyspělým ERP systémem. Svou variabilitou je vhodný i pro firmy s menším počtem zaměstnanců, je však optimalizován především pro středně velké a velké obchodní společnosti.

Tento systém má modulární strukturu, kde každý z modulů tvoří funkční samostatný celek. Mezi moduly patří například prodej a nákup, účetnictví, finance, reklamace, marketing, skladová evidence a další.

3.4 HelpDesk E LINKX a. s.

HelpDesk je služba pro řešení klientských požadavků firmy E LINKX a. s. i interních záležitostí. Všechny takové požadavky mají své zadání, řešitele nebo více řešitelů a stav. Dále lze u požadavků přidávat odhady či procentuální stav, a také přidávat komentáře, což vede k e-mailovému upozornění všech účastníků daného požadavku [8].

3.5 JavaScript

JavaScript [9] objektově orientovaný skriptovací jazyk pro webové stránky, který umožňuje dynamicky aktualizovat obsah, animovat obrázky či ovládat multimédia. Výhodou JavaScriptu je, že běží na straně klienta a nedochází tak k zatěžování prostředků webového serveru. JavaScript patří mezi tři standardní webové technologie, z nichž další dva jsou HTML a CSS.

3.6 Representational State Transfer

REST [10] je architektonický styl pro distribuované prostředí. Je orientován datově, nikoliv procedurálně jako známější XML-RPC nebo SOAP. REST definuje zásady pro vývoj API, využívá protokolu HTTP pro práci se zdroji (angl. resources) v rámci komunikace klient-server.

RESTful je označení pro systémy explicitně využívající metod HTTP k implementaci CRUD operací pro práci se zdroji, které představují data či stavy systému.

3.7 Team Foundation Server

TFS [11] je nástroj společnosti Microsoft poskytující celou sadu nástrojů pro spolupráci při vývoji software. Nabízí nástroje od projektového řízení, přes správu zdrojového kódu až po možnost multiplatformního softwarového řešení. Microsoft také nabízí cloudové řešení službou Visual Studio Team Services (VSTS).

3.8 SoapUI

SoapUI [12] je nástroj pro testování webových služeb SOAP či RESTful nebo služeb založených na protokolu HTTP. Nabízí nejen funkční testování, ale i testování výkonu, zabezpečení či mocking, který umožňuje pomocí falešných objektů simulovat chování těch skutečných.

3.9 SQL Server Management Studio

SSMS [13] je prostředí pro správu libovolné infrastruktury SQL. Toto prostředí zahrnuje širokou sadu grafických nástrojů s řadou editorů skriptů, jak pro vývojáře, tak administrátory. Zajímavostí je zpětná kompatibilita starších verzí. Aktuální verze je již SSMS 17.6 a poskytuje podporu pro nejnovější SQL Server 2017.

4 Testování API rozhraní

V této kapitole se věnuji mé první činnosti, kterou jsem dostal na starost, a to testování API rozhraní. Účelem tohoto API rozhraní je poskytování informací a vykonávání operací pro e-shopy.

Toto API rozhraní je plně RESTful a autentizace je řešena pomocí protokolu OAuth 2.0 [14], který poskytuje mnohem vyšší úroveň zabezpečení oproti jednoduchému ověření přístupu (angl. HTTP Basic Auth) s typem grantu heslo (angl. Password Grant) a formátem dat JSON.

4.1 Zadání

Bylo mi předáno testování API rozhraní pomocí aplikace SoapUI. Mým úkolem bylo nastudovat si dokumentaci ke všem již existujícím metodám, seznámit se s aplikací, vyzkoušet si volání těchto metod a ověřit správnost dat. Následně jsem testoval nově vzniklé metody a dále rozšiřoval dokumentaci. Pokud rozdělím tyto metody na bloky jedná se o:

- Číselníky
- Dodavatelé a odběratelé
- Produkty, produktový katalog a navigátor
- Nabídky, poptávky, objednávky a faktury

4.2 Řešení

Prvním krokem po studiu dokumentace bylo založení nového projektu v aplikaci SoapUI, kde jsem si vytvořil přehledně všechny existující metody. K tomu jsem využil hierarchického uspořádání (stromové struktury). Dále jsem si vytvořil jednotlivé REST požadavky a u každého z nich jsem nastavil typ metody, koncový bod (angl. endpoint), zdroj (angl. resource), parametry v případě GET metod a JSON strukturu požadavku v případě POST metod. Nakonec bylo třeba ještě nastavit oprávnění pro autentizaci.

K testování jsem dostával zejména dva typy metod, a to GET a POST. Metody POST byly dále ještě rozděleny na synchronní (online) a asynchronní. Synchronní metody umožnují zaslat pouze jeden objekt a výsledek požadavku vrací ihned, v případě platného požadavku obsahuje odpověď stavový kód 201 Created. Asynchronní metody umožnují zaslat pole objektů a v případě platného požadavku je vygenerován GUID, který je vracen v odpovědi se stavovým kódem 202 Accepted. Ke zpracování dat dochází automatem, který se spouští například 1x za hodinu. Na výsledek požadavku se bylo nutné doptat pomocí GET požadavku s parametrem GUID. Odpovědí je pole objektů, které obsahují aktuální stav, čas zpracování, popřípadě chybový kód a zprávu.

4.2.1 Metody GET

Metody GET slouží k získání dat jako jsou například číselníky, seznam dodavatelů či odběratelů nebo produktová karta. S GET metodami se také pojí filtrování dat. To je realizováno v aplikaci

pomocí tabulky parametrů. Z té se následně vytvoří QueryString, který je parametrem požadavku. Ověřoval jsem, jestli filtrování pomocí parametrů vrací odpovídající data, a také zdali zobrazená data odpovídají těm v databázi využitím dotazů nebo aplikace Esyco .NET.

Nejčastějším problémem byla neočekávaná chyba serveru, kdy odpověď obsahovala stavový kód 500 Internal Server Error, což byl problém neošetřených výjimek. Další nejčastější chybou byly nekompletní objekty, kdy programátor zapomněl na některý atribut entity. Posledním nejčastějším problémem byla nedostupnost zdroje, kdy odpověď obsahovala stavový kód 404 Not Found. Většinou se požadovaná metoda nacházela na jiném zdroji nebo ještě nebyla nasazena na API rozhraní.

Ukázka prostředí aplikace SoapUI a volání metody GET včetně odpovědi je součástí elektronické přílohy v adresáři:

`./04-TestovaniAPI/Ukazka-prostredi-aplikace-SoapUI.png`

4.2.2 Metody POST

Metody POST slouží k vytváření dat jako jsou například založení dodavatele či odběratele nebo vytváření nabídek, poptávek a objednávek. U těchto metod jsem ověřoval nejen jestli proběhne vytvoření, ale také jestli vytvořená entita nese všechny atributy, které byly zaslány v požadavku. Dále jsem rovněž ověřoval správnost povinnosti objektů (páry název-hodnota), zdali jsou v daném požadavku povinné či nikoliv.

Nejčastější chybou byly nekompletní entity, u nichž chyběly některé atributy. Problém byl způsoben tím, že zpracování požadavků z API rozhraní do importních tabulek mělo na starost jiné oddělení než zpravování dat z importních tabulek a následné vytvoření entit.

Další nejčastější chybou byla také neočekávaná chyba serveru, kdy odpověď obsahovala stavový kód 500 Internal Server Error, což byl opět problém neošetřených výjimek.

4.2.3 Dokumentace

Dokumentace všech metod byla klíčová nejen pro účely testování, ale taky jako výchozí bod manuálu pro uživatele. Ukázka dokumentace je součástí elektronické přílohy v adresáři:

`./04-TestovaniAPI/Ukazka_dokumentace_API.docx`

Práce	Hodin
Testování	26,6
Školení/konzultace	5,7
Psaní dokumentace/helpů	3,8
Hodin celkem	**36,1**

5 Automatizace testů API rozhraní

V této kapitole se věnuji automatizaci testů pro API rozhraní z předchozí kapitoly. Jak jsem již zmiňoval, do té doby jsem s nimi neměl žádnou zkušenost. Se vzrůstajícím počtem nových metod a zákazníků jsem dostal za úkol prvotní vývoj automatizovaných testů. Přibližně se jednalo asi o 70 metod a 20 koncových bodů, kde na každém z nich bylo potřeba testovat různý počet těchto metod.

5.1 Zadání

Vedoucím webového oddělení mi byla předána vize o automatizovaných testech. Jako ukázku jsem dostal projekt, který sice obsahoval automatizované testy, avšak jednalo se o testy na webovou službu založenou na XML SOAP. Pro větší rozhled jsem byl odkázán na dokumentaci aplikace SoapUI.

5.2 Řešení

Prvním krokem bylo prostudování dokumentace aplikace SoapUI a rozdělení metod na logické celky. Začal jsem nejjednodušším celkem, a to přibližně 15 číselníky.

5.2.1 Testovací sada, případ a krok

V SoapUI jsem si vytvořil novou testovací sadu (angl. TestSuite). Do této sady jsem si dále vytvořil nový testovací případ (angl. TestCase), který představuje logický celek. Ten se skládá z testovacích kroků (angl. TestStep), což jsou jednotlivé REST požadavky, ale i vlastnosti a jejich přenos, skoky, volání, skripty, zpoždění, manuální krok a další.

5.2.2 Autentizace

Po vytvoření kroků pro získání všech číselníků jsem došel k problému s autentizací. Proces autentizace u OAuth 2.0 Password Grant probíhá metodou POST o přístupový token na autorizační server. Tělo požadavku má formát prostého textu (angl. plain text). Celý požadavek vypadá následovně:

```
POST https://esycoapi-test.edsystem.cz/oauth/token HTTP/1.1


grant_type=password
&username=USERNAME
&password=PASSWORD
&client_id=CLIENT_ID
&client_secret=CLIENT_SECRET
```

Je-li požadavek platný, autorizační server vygeneruje a vrátí přístupový token, typ a platnost tokenu a token pro obnovení. Odpověď autorizačního serveru vypadá následovně:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token":"GWGaTc1uBtd33Cro88cmKgwWrxOCmKKS",
  "token_type":"bearer",
  "expires_in":7200,
  "refresh_token":"ba3b735f51864b18b2a8ecd06df1835eTV"
}
```

S tímto přístupovým tokenem lze posílat ověřené požadavky na rozhraní API. Do HTTP hlavičky každého požadavku stačí přidat následující parametr s hodnotou:

```
Authorization: Bearer GWGaTc1uBtd33Cro88cmKgwWrxOCmKKS
```

Proces autentizace tedy bylo třeba zahrnout do testovacího případu a výsledný přístupový token si uložit do vlastností a jednoduchým skriptem ho rozdistribuovat do hlaviček každého požadavku. Skripty lze psát v SoapUI pomocí programovacího jazyka Groovy a skriptovacího jazyka JavaScript. Posledním krokem k vytvoření prvního testovacího případu bylo nastavit tvrzení (angl. assertions) u každého kroku pro kontrolu odpovědi požadavku. Lze ověřovat stavový kód, jestli tělo odpovědi obsahuje konkrétní řetězec nebo odpovídá konkrétnímu schématu a další. V tomto případě jsem nastavil jen ověření úspěšného stavového kódu, a to 200 OK v případě GET metod.

5.2.3 **Skripty**

U další testovací sady na celek dodavatelé jsem se potýkal s problémem, jak seřadit jednotlivé kroky za sebe a odkud získat potřebné parametry. Začal jsem tedy požadavkem na seznam dodavatelů. Pomocí skriptu jsem z odpovědi vybral jednoho z dodavatelů a přenesl jeho identifikaci (id) do požadavku pro detail dodavatele a požadavku na seznam adres dodavatele. Z odpovědi na seznam adres jsem také pomocí dalšího skriptu vybral jednu z adres a její id společně s id dodavatele přenesl do posledního požadavku na detail adresy. Také u těchto metod jsem nastavil ověření úspěšného stavového kódu 200 OK a zároveň zdali tělo odpovědi (objekt či pole objektů) obsahuje objekt s názvem id pomocí JSONPath výrazu. Synchronní a asynchronní POST

K dalšímu problému jsem se dostal u testovací sady na celek objednávky pro koncové zákazníky (B2C). Tato sada zahrnovala vytvoření objednávky B2C pomocí metod POST – synchronní (online) a asynchronní. Rozdíly jsem popisovat již v předchozí kapitole 4.2. Posledním krokem bylo ověření vytvoření obou objednávek pomocí metody GET pro získání detailu objednávky B2C, na základě id objednávky. Bylo třeba vyřešit, jak získat id z odpovědí těchto metod. U asynchronní metody to bylo značně složitější, jelikož nedošlo ke zpracování dat a výsledkem bylo jen získaní GUID.

Za synchronní (online) vytvoření objednávky B2C jsem vložil nový skript, pomocí kterého jsem z HTTP hlavičky odpovědi získal adresu pro přesměrování (angl. location) na detail vytvořené objednávky B2C.

```
Location: http://esycoapi-test.edsystem.cz/v1/clients/orders/customer/b2c/123456
```

Pokud se podíváme na strukturu adresy, zjistíme že poslední část za lomítkem tvoří id vytvořené objednávky B2C. Výsledný skript pro získání id byl následovný:

```
import net.sf.json.*;

import net.sf.json.groovy.*;

def location = testRunner.testCase.testSteps['ImportB2COrder - online'].testRequest.response.responseHeaders['Location'][0];

def sp_location = location.split('/');

def id = sp_location[sp_location.length-1];

testRunner.testCase.testSteps['Properties'].setPropertyValue("orderId", id.toString() );
```

U asynchronního volání jsem problém vyřešil tak, že za volání metody jsem vložil jednoduchý skript, který přenesl GUID do GET metody pro získání výsledku požadavku. Za tento skript jsem dále vložil manuální krok, který informuje uživatele o tom, že je třeba ručně spustit zpracování dat automatem. Po doběhnutí bylo třeba odkliknout manuální krok, aby test pokračoval dalším krokem – získáním výsledku požadavku dle GUID v parametru. Z pole objektů odpovědí jsem dalším scriptem vytáhl již id vytvořené objednávky a přenesl jej do metody GET pro získání detailu objednávky. Také u této metody jsem nastavil ověření úspěšného stavového kódu 200 OK a zároveň zdali tělo odpovědi obsahuje objekt s názvem id pomocí JSONPath výrazu.

Práce	**Hodin**
Programování	30,8
Testování	19,8
Školení/konzultace	4,1
Psaní dokumentace/helpů	4,2
Hodin celkem	**58,9**

6 Projekt MailChimp

V této kapitole se věnuji stěžejnímu bodu práce projektu MailChimp, jehož cílem je napojení ERP systému Esyco .NET na službu MailChimp skrze rozhraní API. U tohoto projektu jsem dostal na starost celou část implementace. Nejdříve však bylo třeba vypracovat datovou analýzy a návrh implementace včetně stanovení časových odhadů na základě analýzy a návrhu. Analýzu společně s testováním aplikace dostala na starost kolegyně a vedoucím projektu byla projektová manažerka, která koordinovala celý průběh.

6.1 Zadání

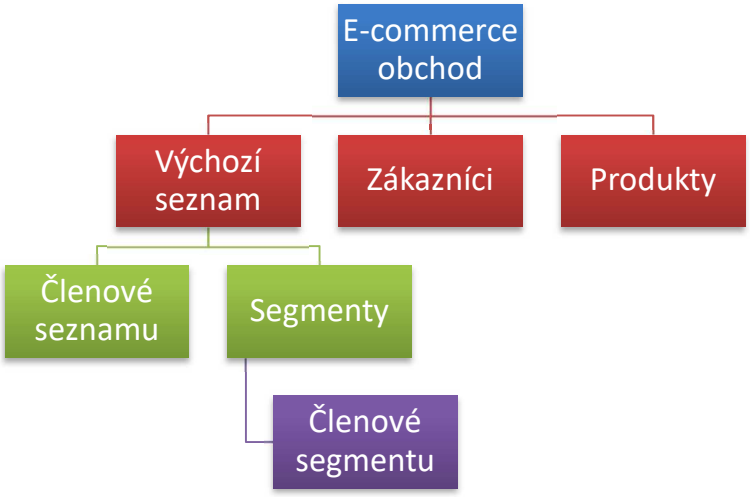
Zadáním projektu bylo napojení ERP systému Esyco .NET na službu MailChimp. Dále vypracování analýzy, návrhu komplexního využití možností vzhledem k datům, které lze využít z ERP systému, návrhu technického řešení napojení s využitím API metod MailChimpu a provedení samotné implementace.

6.2 Řešení

První část měla na starost kolegyně, a proto jsem se mezitím věnovat testování API rozhraní a automatizaci těchto testů, které jsem popisoval v předchozích dvou kapitolách. Později jsem se však do analýzy přece jen zapojil s cílem vypracovat datovou analýzu jak z pohledu služby MailChimp, tak z pohledu IS Esyco .NET.

6.2.1 Datová analýza a návrh implementace

K datové analýze jsem dostal k dispozici analýzu, ze které jsem si nejprve zjistil, s jakými daty budu vůbec pracovat. Jednalo se o E-commerce obchod, zákazníky, seznamy a jejich členy, segmenty a jejich členy. Dalším krokem bylo určit si a definovat vazby mezi těmito objekty, což jsem řešil sadou menších diagramů. Pokud bych to však poskládal do jednoho diagramu, vypadal by následovně.



Obrázek 1.1: *2 Diagram vazeb objektů služby MailChimp*

Dalším krokem po určení a definování vazeb mezi objekty bylo definovat, co tyto objekty znamenají z pohledu dat IS Esyco .NET. Tuto část jsem vyřešil pomocí tabulek, kde jsem si vypsal parametry objektu a k němu navázal data z IS Esyco .NET. Když už jsem znal to, jaká data budu přenášet z IS Esyco .NET, bylo také třeba si vymezit pravidla pro přenos těchto dat, respektive jejich restrikci.

Po tomto kroku jsem mohl přejít k samotnému návrhu implementace s cílem vytvořit jakousi abstrakci zdrojového kódu, která bude sloužit jako výchozí boj při implementační fázi. Začal jsem tedy sestavením obecného třídního diagramu. Dále jsem pokračoval stavovým diagramem, u kterého jsem zjistil, že všechny tyto objekty mají stejnou množinu stavů – CRUD operace.

Na základě těchto skutečností jsem se rozhodl pro konzolovou aplikaci na platformě .NET Framework, která se bude spouštět jako služba na příslušné databázi ve stanovených časových intervalech. Pro přípravu dat jsem navrhl buď použití pohledů nebo uložených procedur, které by plnily dočasné tabulky nebo tabulky. Výběr jsem si ponechal až do implementace otevřený. V této chvíli jsem si nebyl jistý, jestli budu potřebovat s daty dále pracovat nebo ne, a také zdali budu zpracovávat a zapisovat odpovědi z API rozhraní. Pro přístup ke zdrojům dat na SQL Serveru jsem zvolil technologii ADO.NET.

Posledním krokem bylo rozdělení projektu na fáze, vymezení těchto fází a vytvoření časového harmonogramu včetně stanovení odhadů a termínů. Ukázka datové analýzy a návrhu implementace je součástí elektronické přílohy v adresáři:

`./08-MailChimp/Ukazka-navrh-implementace.docx`

Pro orientaci v implementaci jednotlivých fází slouží třídní diagram. Ten je součástí elektronické přílohy v adresáři:

`./08-MailChimp/Tridni-diagram.png`

6.2.2 **Implementace**

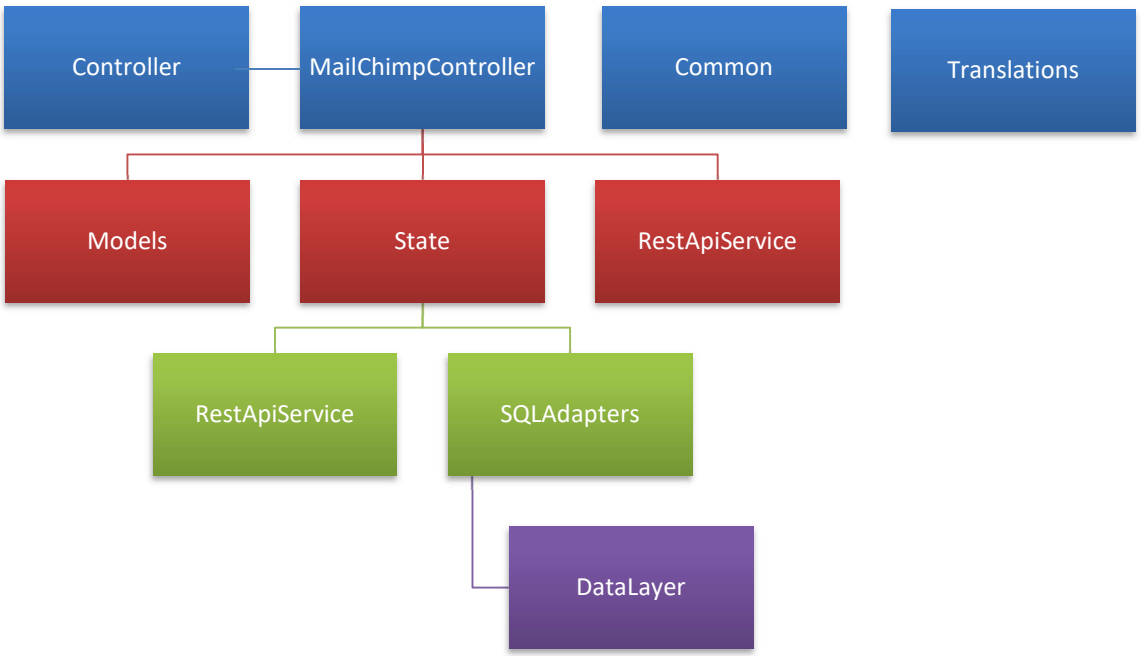
Úkolem 1. fáze bylo naimplementovat základní strukturu aplikace a REST API službu pro komunikaci s API MailChimpu. Další krok zahrnoval import a synchronizaci klientů (subjektů) se službou MailChimp. Ten zahrnoval implementaci nových metod REST API služby, přípravu dat i samotnou synchronizaci.

Prvním krokem bylo vytvoření nového projektu konzolové aplikace (.NET Framework). V tomto projektu jsem si rozvrhl základní strukturu aplikace. Základem byla architektura MVC s využitím modelu (angl. model) a řadiče (angl. controller). Pohled (angl. view) nebylo potřeba z prostého důvodu, a to že aplikace přestavuje službu, kde API rozhraní služby MailChimp je server a tato aplikace je klient, založenou na architektuře klient-server.

Dle třídního diagramu si můžeme všimnout, že základní strukturu aplikace tvoří devět komponent. Mezi tyto komponenty patří:

- Common – zahrnuje skupinu společných tříd pro celý projekt. Příkladem mohou být globály (angl. Globals), které představují sadu globálních parametrů důležitých pro běh aplikace.
- Controllers – jedná se o řadiče, které řeší řídicí logiku aplikace. Aplikace má tyto řadiče rovnou dva. Jeden pro obecné řízení logiky aplikace a druhý pro řízení logiky komunikace s API rozhraním služby MailChimp.
- DataLayer – zahrnuje třídu, která zprostředkovává přístup ke zdrojům dat na SQL Serveru využívající technologii ADO.NET.
- DataMapper – představuje vrstvu pro mapovaní dat mezi objekty databáze a objekty v paměti. Pojem Data Mapper označuje architektonický vzor pro práci s datovými zdroji.
- Models – představuje datovou strukturu realizovanou pomocí tříd, což jsou objekty, s nimiž aplikace pracuje.
- RestApiService – jedná se o sadu služeb, které přímo komunikují s API rozhraním služby MailChimp pomocí třídy HttpClient.
- SQLAdapters – je vrstva, která zajišťuje komunikaci mezi aplikací a datovou vrstvou. Pojem adaptér (angl. adapter) označuje návrhový vzor, který převádí rozhraní jedné třídy do jiného rozhraní.
- State – zapouzdřuje chování objektů na základě stavů. Pojem stav (angl. state) opět není náhodný, ale označuje stavový vzor (angl. state pattern).
- Translations – je samostatná komponenta, jež poskytuje jednoduché řešení překladů pro logování.

Vazby mezi těmito komponenty popisuje následujícím diagram.



Obrázek 1.2: *3 Diagram vazeb komponent aplikace*

V 1. fázi jsem tedy implementoval komponenty Controller, MailChimpController, Models, RestApiService, SQLAdapters, DataLayer a Common. Komponenta State vznikla až během implementace dalších fází s cílem přesunout logiku procesu synchronizace mimo MailChimpController. Poslední komponenta Translations vznikla společně s nutností logování, které jsem řešil později v rámci Controlleru.

Třída Controller tedy řídí logiku aplikace jako inicializaci komponent při spuštění, logování či běh aplikace v určitém módu. Tato třída je realizována pomocí návrhového vzoru jedináček (angl. singleton) pro zajištění pouze jedné instance v celém programu. Mód aplikace i další parametry se definují spouštěcími argumenty z příkazové řádky, která obsahuje i nápovědu. Ukázka příkazové řádky je součástí elektronické přílohy v adresáři:

`./08-MailChimp/Ukazka-prikazove-radky.png`

S třídou MailChimpController přímo komunikuje Controller, který volá spuštění procesu synchronizace dle módu. MailChimpController má za úkol řízení logiky komunikace s API rozhraním služby MailChimp například inicializaci spojení a spuštění procesu synchronizace jednotlivých objektů služby MailChimp. Klíčovou vlastností pro běh synchronizace je prvotní inicializaci E-commerce obchodů a jejich seznamů.

Models představuje objekty služby MailChimp. V 1. fázi jsem tedy implementoval objekty E-commerce obchod, seznam, zákazník, člen, adresa a chyba. U objektů adres jsem využil společných rysů pro jednotlivě adresy pomocí dědičnosti. Speciálním objektem jsou kolekce, které obsahují pole objektů. K vytvoření všech těchto objektů jsem použil nástroj json2csharp [15], který na využívá JSON schéma objektu k vygenerování třídy včetně proměnných, přístupových metod i komentářů. Toto řešení mi však přišlo zbytečně složité, tak jsem tyto proměnné upravil na vlastnosti a přístup k nim jsem zajistil pomocí gettetů (navrácení hodnoty) a setterů (zápis hodnoty). Ukázka, jak taková výsledná třída vypadá (E-commerce obchod), je součástí elektronické přílohy v adresáři:

`./08-MailChimp/Store.cs`

Komponenta State vznikla až v dalších fázích implementace s cílem přesunout logiku procesu synchronizace mimo MailChimpController. K tomu jsem využil stavový vzor, který zapouzdřuje chování objektu na základě jeho stavu. Základem je abstraktní třída AbstractState z niž dědí třídy objektů, které je potřeba synchronizovat. Ty pak implementují metody pro stavy objektů. Ukázka třídy AbstractState je součástí elektronické přílohy v adresáři:

`./08-MailChimp/AbstractState.cs`

RestApiService zahrnuje sadu služeb, které přímo komunikují s API rozhraním služby MailChimp pomocí třídy HttpClient. Jako podklad pro implementaci jsem použil téma o volání webového rozhraní API z klienta .NET [16], které patří mezi pokročilá témata z dokumentace Microsoftu.

Pro přípravu dat jsem se nakonec rozhodl pro uložené procedury, které budou plnit tabulky. Procedury se tedy starají o přípravu dat do tabulek. Každý záznam má svou akci, stav, časové razítko pro vytvoření a synchronizaci, popřípadě chybový kód a zprávu. Aplikace pak postupně z těchto dat synchronizuje tyto objekty s API rozhraním služby MailChimp. Výsledný stav zapíše po zpracování odpovědi k danému záznamu. Relační diagram je součástí elektronické přílohy v adresáři:

`./08-MailChimp/Relacni-diagram.png`

Seznam tabulek a jejich krátký popis:

- IS_MAILCHIMP_CDL_REG – tabulka pro E-commerce store,
- IS_MAILCHIMP_SUBJ – tabulka pro zákazníky,
- IS_MAILCHIMP_DIST_LIST_HEAD – tabulka pro hlavičky distribučních seznamů,
- IS_MAILCHIMP_ DIST_LIST – tabulka pro členy distribučních seznamů,
- IS_MAILCHIMP_MAR_SRT_HEAD – tabulka pro hlavičky marketingových zařazení,
- IS_MAILCHIMP_ MAR_SRT – tabulka pro členy marketingových zařazení,
- IS_MAILCHIMP_PRO – tabulka pro produkty,
- IS_MAILCHIMP_REG_BND_PRO – tabulka pro vazbu mezi E-commerce obchody a produkty,
- IS_MAILCHIMP_CDL_ACT – číselník pro akce (CRUD operace),
- IS_MAILCHIMP_CDL_STATUS – číselník pro stavy.

Seznam procedur a jejich krátký popis:

- PR_MAILCHIMP_SubjPrep – procedura pro přípravu zákazníků,
- PR_MAILCHIMP_DistListPrep – procedura pro přípravu distribučních seznamů (hlaviček i členů),
- PR_MAILCHIMP_MarSrtPrep – procedura pro přípravu marketingových zařazení (hlaviček i členů),
- PR_MAILCHIMP_ProPrep – procedura pro přípravu zákazníků.

Úkolem 2. fáze bylo zajištění importu a synchronizace distribučních seznamů a marketingových zařazení se službou MailChimp, včetně segmentování klientů (subjektů) z předchozí fáze. V této fázi se zároveň řešilo i zpracování odpovědí ze API MailChimpu.

Bylo tedy nutné rozšířit Models o objekt segmenty, který zahrnoval jak distribuční seznam, tak marketingové zařazení, a také jsem doplnil RestApiService o další službu. Pomocí dvou procedur jsem zajistil přípravu dat nejen pro tyto segmenty, ale i pro klienty těchto segmentů a následně jsem implementoval samotnou synchronizaci přidáním dalšího objektu do State. Také

jsem se zde věnoval zpracování požadavku, protože bylo třeba uložit id vytvářených segmentů pro další synchronizaci.

Úkolem 3. fáze bylo zajištění importu a synchronizace produktů se službou MailChimp. Tato fáze však zahrnovala nejen produkty, ale i jejich regiony, do kterých jsou produkty rozděleny.

Bylo tedy nutné rozšířit Models o objekty produkt, varianta, obrázek, a také jsem doplnil RestApiService o další službu. Pomocí procedury jsem zajistil přípravu dat pro tyto produkty a následně jsem implementoval samotnou synchronizaci přidáním dalšího objektu do State. Také jsem se zde věnoval komponentě Translations, která vznikla společně s nutností logování. Pro logování jsem využil knihovny log4net [17].

6.2.3 **Dokončovací práce**

Cílem těchto prací bylo finální dokončení projektu. Důraz se kladl hlavně na komentáře v kódu, ale i na vylepšování a čištění kódu aplikace tzv. refaktorování (angl. refaktoring). Taktéž bylo třeba vypracování technické dokumentace, uživatelského manuálu a nasazení do testovacího provozu.

Technická dokumentace je klíčová pro zprovoznění aplikace. Informuje o tom, co vše je nutné nastavit, a také jakým způsobem, aby aplikace fungovala. Ukázka technické dokumentace je součástí elektronické přílohy v adresáři:

`./06-MailChimp/Ukazka_technicke_dokumentace.docx`

Uživatelský manuál informuje o tom, jaké data se přenáší do služby MailChimp, co musí v IS Ecyco .NET nastavit, a také základní práce s GUI služby MailChimp. Ukázka uživatelského manuálu je součástí elektronické přílohy v adresáři:

`./06-MailChimp/Ukazka_manual.docx`

Práce	**Hodin**
Programování	226,4
Analýza	20,0
Testování	0,2
Školení/konzultace	19,1
Psaní dokumentace/helpů	8,5
Hodin celkem	**274,2**

7 Projekt SmartEmailing

Jelikož systémů na e-mail marketing je opravdu mnoho, je důležité dát zákazníkům na výběr. V této kapitole se tedy věnuji projektu SmartEmailing. V tomto projektu je mým cílem analytická činnost. Implementační část a její návrh dostala na starost kolegyně a vedoucím projektu byla opět projektová manažerka.

7.1 Zadání

Zadáním projektu bylo napojení ERP systému Esyco .NET na službu SmartEmaling. Dále vypracování analýzy, návrhu komplexního využití možností vzhledem k datům, které lze využít z ERP systému, návrhu technického řešení napojení s využitím API metod SmartEmalingu a provedení samotné implementace.

7.2 Řešení

V rámci práce jsem se rozhodl analýzu nerozepisovat, ale přiložit pouze její ukázku a časovou náročnost. Ukázka je součástí elektronické přílohy v adresáři:

`./07-SmartEmailing/Ukazka_analyza.docx`

Práce	Hodin
Analýza	31,5
Školení/konzultace	6,2
Hodin celkem	**37,7**

8 Zhodnocení

8.1 Získané odborné znalostí a dovednosti ze studia a jejich využití

Díky znalostem získaným ze studia na VŠB-TU v Ostravě v oblasti programování a databází jsem byl dostatečně připraven na přijímací pohovor, který jsem musel podstoupit a vstupní test, který jsem musel vypracovat již v předchozím ročníku v předmětu Praxe. Znalosti se týkaly předmětů Algoritmy I a II programovací jazyky I a II, a také Úvod do databázových systémů a Databázové a informační systémy. Informace, které jsem z těchto předmětů získal, byly nedílnou součástí práce.

8.2 Chybějící odborné znalosti a dovednosti

V průběhu praxe jsem musel čelit neznámým věcem, které jsem doposud neznal. Byla to například práce s logováním, práce s HttpClient nebo asynchronní programování. Nicméně mě neznalost těchto záležitostí neodradila, jelikož jsem byl připraven učit se novým věcem, které bylo nutné znát, abych dokázal řešit dané úkoly.

Závěr

V průběhu odborné praxe jsem si prošel všemi stádii vývoje software. Při testování API rozhraní jsem přišel na to, co vše tento proces zahrnuje a jak důležité je. Rovněž jsem pochopil princip, jak samotné testy tvořit nebo jak je lze zautomatizovat. V průběhu mi došlo i to, jak klíčová je analýza pro návrh implementace a jak složité je vypracovat tento návrh či dokonce stanovit časové odhady.

Dále jsem se ponořil i do implementace aplikace, kde opravdu oceňuji dlouholeté zkušenosti programátorů, kteří mi byli k dispozici ke konzultacím. Výbornou zkušeností také bylo vyzkoušet si vodopádovou metodiku vývoje software, ale později také i agilní metodiku. V neposlední řadě jsem si také zkusil i analytickou činnost a zjistil, jak tvořit business analýzu, jak vypadá její struktura a jak klíčová je pro další vývoj.

Odbornou praxi ve společnosti E LINKX a. s. a nejen ji můžu doporučit všem, kteří chtějí obohatit své praktické dovednosti, získat přehled o pracovních pozicích a vyzkoušet si je. Dále také pracovat v přátelském prostředí a zažít práci nejen v týmu, ale i samostatně. Je to určitě první krok vpřed, jak se připravit na kariéru v IT oboru.

Použitá literatura

[1] O nás. E LINKX a. s. [online]. [cit. 2018-04-15]. Dostupné z: http://elinkx.cz/o-nas/

[2] C# 6.0 draft specification. Microsoft Docs [online]. [cit. 2018-04-15]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction

[3] What is .NET?. Microsoft [online]. [cit. 2018-04-15]. Dostupné z: https://www.microsoft.com/net/learn/what-is-dotnet

[4] .NET Framework Guide. Microsoft Docs [online]. [cit. 2018-04-15]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/

[5] MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 1. Brno: Zoner Press, 2011. Encyklopedie Zoner Press. ISBN 978-80-7413-131-8.

[6] MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 2. Brno: Zoner Press, 2011. Encyklopedie Zoner Press. ISBN 978-80-7413-145-5.

[7] Esyco business. E LINKX a. s. [online]. [cit. 2018-04-15]. Dostupné z: http://elinkx.cz/esyco-business/

[8] HelpDesk E LINKX a. s. [online]. [cit. 2018-04-15]. Dostupné z: http://helpdesk.elinkx.cz

[9] About JavaScript. JavaScript | MDN [online]. [cit. 2018-04-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[10] What is REST?. Codecademy [online]. [cit. 2018-04-15]. Dostupné z: https://www.codecademy.com/articles/what-is-rest

[11] Team Foundation Server. Visual Studio [online]. [cit. 2018-04-15]. Dostupné z: https://www.visualstudio.com/cs/tfs/

[12] Getting Started With SoapUI. SoapUI by SmartBear [online]. [cit. 2018-04-15]. Dostupné z: https://www.soapui.org/getting-started/introduction.html

[13] SQL Server Management Studio (SSMS). Microsoft Docs [online]. [cit. 2018-04-15]. Dostupné z: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms

[14] OAuth 2.0. OAuth Community Site [online]. [cit. 2018-04-15]. Dostupné z: https://oauth.net/2/

[15] JSON Schema to C #. JSON Schema to C # Converter – Online [online]. [cit. 2018-04-20]. Dostupné z: http://json2csharp.rohitl.com/

[16] Call a Web API From a .NET Client (C#). Microsoft Docs [online]. [cit. 2018-04-20]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/web-api/overview/advanced/calling-a-web-api-from-a-net-client

[17] Apache log4net. Apache Logging Services [online]. [cit. 2018-04-20]. Dostupné z: https://logging.apache.org/log4net/

Seznam příloh

Součástí BP je CD.

Adresářová struktura přiloženého CD/DVD: